IN4393-16 Computer Vision - Q3
Lab Exercise 4
Optical Flow & Structure from Motion

April 6, 2018

Students should work on this lab exercise in groups of two people. The assignment should be completed before 23th April 2018.

1 Optical Flow

1.1 Introduction

Optical flow is the apparent motion of image pixels or regions from one frame to the next, which results from moving objects in the image or from camera motion. Underlying optical flow is typically an assumption of brightness constancy; that is, the image values (brightness, colour, etc) remain constant over time, though their 2D position in the image may change. Algorithms for estimating optical flow exploit this assumption in various ways to compute a velocity field that describes the horizontal and vertical motion of every pixel in the image. For a 2D+t dimensional case a voxel at location $(x; y; t)$ with intensity $I(x; y; t)$ will have moved by δ_x, δ_y and δ_t between the two image frames, and the following image constraint equation can be given:

$$I(x, y, t) = I(x + \delta_x, y + \delta_y, t + \delta_t) \tag{1}$$

Assuming the movement to be small, the image constraint at I(x, y, t) with Taylor series can be developed to get:

$$I(x+\delta_x, y+\delta_y, t+\delta_t) = I(x,y,t) + \frac{\partial I}{\partial x}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t + h.o.t. \quad (2)$$

As we assume the image values remain constant over time, we will get:

$$\frac{\partial I}{\partial x}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t = 0 \quad (3)$$

or equivalently

$$I_x V_x + I_y V_y = -I_t \quad (4)$$

where V_x, V_y are the x and y components of the velocity or optical flow of $I(x;y;t)$, and I_x, I_y and I_t are the derivatives of the image at $(x;y;t)$ in the corresponding directions. This is the main equation of the optical flow (https://en.wikipedia.org/wiki/Optical_flow).

Optical flow is hard to compute for two main reasons. First, in image regions that are roughly homogeneous, the optical flow is ambiguous because the brightness constancy assumption is satisfied by many different motions. Second, in real scenes the assumption is violated at motion boundaries and by changing lighting, non-rigid motions, shadows, transparency, reflections, etc. To address the former, all optical flow methods make some sort of assumption about the spatial variation of the optical flow that is used to resolve the ambiguity; these are just assumptions about the world which will be approximate and consequently may lead to errors in the flow estimates. The latter problem can be addressed by making much richer but more complicated assumptions about the changing image brightness or, more commonly, using robust statistical methods which can deal with 'violations' of the brightness constancy assumption.

1.2 Lucas-Kanade Algorithm

We will be implementing the Lucas-Kanade method for Optical Flow estimation. This method assumes that the optical flow is essentially constant in a local neighbourhood of the pixel under consideration. Therefore, the main equation of the optical flow can be assumed to hold for all pixels within a window cantered at the pixel under consideration. Lets consider pixel p then for all pixels around p the local image flow vector $(V_x; V_y)$ must satisfy

$$
\begin{aligned}
I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\
I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\
&\vdots \\
I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n)
\end{aligned} \tag{5}
$$

where $q_1, q_2, ...q_n$ are the pixels inside the window around p, and $Ix(q_i)$, $Iy(q_i)$, $It(q_i)$ are the partial derivatives of the image I with respect to position x, y and time t, evaluated at the point q_i and at the current time.

These equations can be written in matrix form $Av = b$, where

$$
A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_x(q_n) \end{bmatrix}, v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \tag{6}
$$

This system has more equations than unknowns and thus it is usually overdetermined. The Lucas-Kanade method obtains a least square solution as

$$
v = (A^T A)^{-1} A^T b \tag{8}
$$

For this assignment, you will be given two pairs of images: $synth1.pgm$, $synth2.pgm$; and $sphere1.ppm$, $sphere2.ppm$. You should estimate the optical flow between these two pairs; it means you will get optical flow for sphere images, and for synth images separately.

You can use regions that are 15×15 pixels, and non-overlapping, i.e., if the input images are 200×200, you should have an array of 13×13 optical flow vectors at the end of your procedure. As we consider 15×15 regions, your matrix A will have the following size 225×2, and the vector b will be 225×1.

The algorithm can be summarized as follows:

1. Divide input images on non-overlapping regions, each region being 15 $\times$ 15 pixels.

2. For each region compute A, A^T and b; then estimate optical flow as given in equation (8).

3. When you have estimation for optical flow $(V_x; V_y)$ of each region, you should display the results. There is a Matlab function quiver which plots a set of two-dimensional vectors as arrows on the screen. Try to figure out how to use this to plot your optical flow results. An example plot is illustrated in Figure [1].

(a) synth (b) sphere

Figure 1: quiver plot

1.3 Tracking

In this part you will implement a simple tracking algorithm and test it on *model_house* data as follows:

1. Download *model_house* data file, which includes all the 101 images and a measurement matrix consisting of 215 points visible in each of the 101 frames (see readme file inside archive for details). Use the points given in the **measurement_matrix.txt** on the first frame.

2. Then, track these points with Lucas-Kanade method for optical flow estimation.

3. Compare your results with the given points on each frame. Plot the per-frame error (sum of squared Euclidean distances, in pixels, between the found and given points) as a function of the frame number. Prepare a video to visualize the initial feature points and the flow (including the given points). A sample video is uploaded in Brightspace as a reference.

2 Structure from Motion

Now, use the feature points found by your implementation as input for the affine structure from motion procedure described in your lecture notes. Remember to enable a sufficient number of points that persist throughout the sequence to perform the factorization on a dense matrix. There is no need to fill in missing data for this problem. Use the following scheme:

1. Normalize the point coordinates by translating them to the mean of the points in each view (see lecture for details).

2. Apply SVD to the $2M \times N$ data matrix to express it as $D = U * W * V'$ where U is a $2M \times 3$ matrix, W is a 3×3 matrix of the top three singular values, and V is a $N \times 3$ matrix. Derive structure and motion matrices from the SVD as explained in the lecture.

3. Use plot3 to display the 3D structure. Discuss whether or not the reconstruction has an ambiguity.

4. Repeat these steps using the given points in the **measurement_matrix-.txt** and compare the results with those of the tracked points.